[illegible]

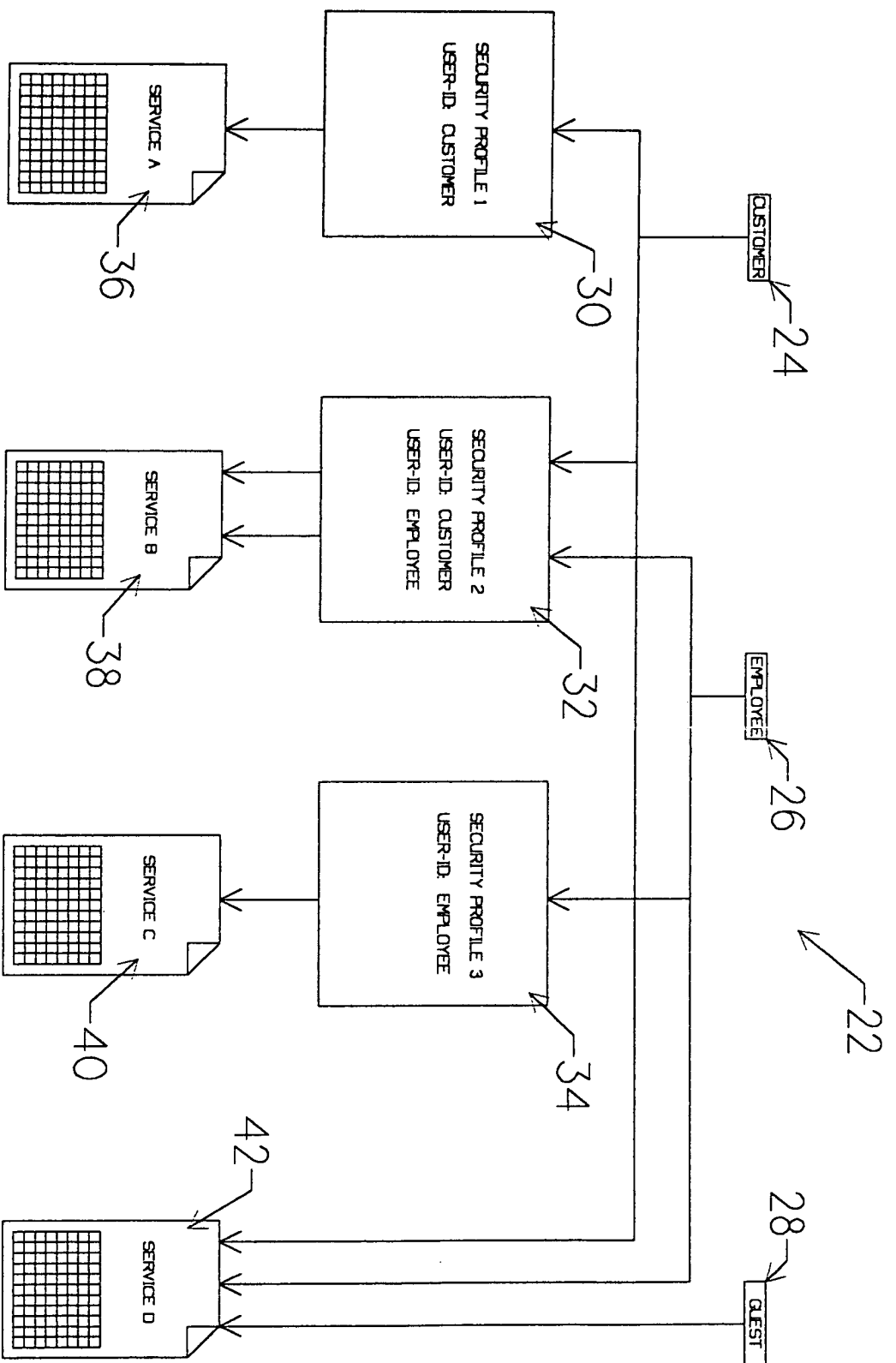


FIG. 2

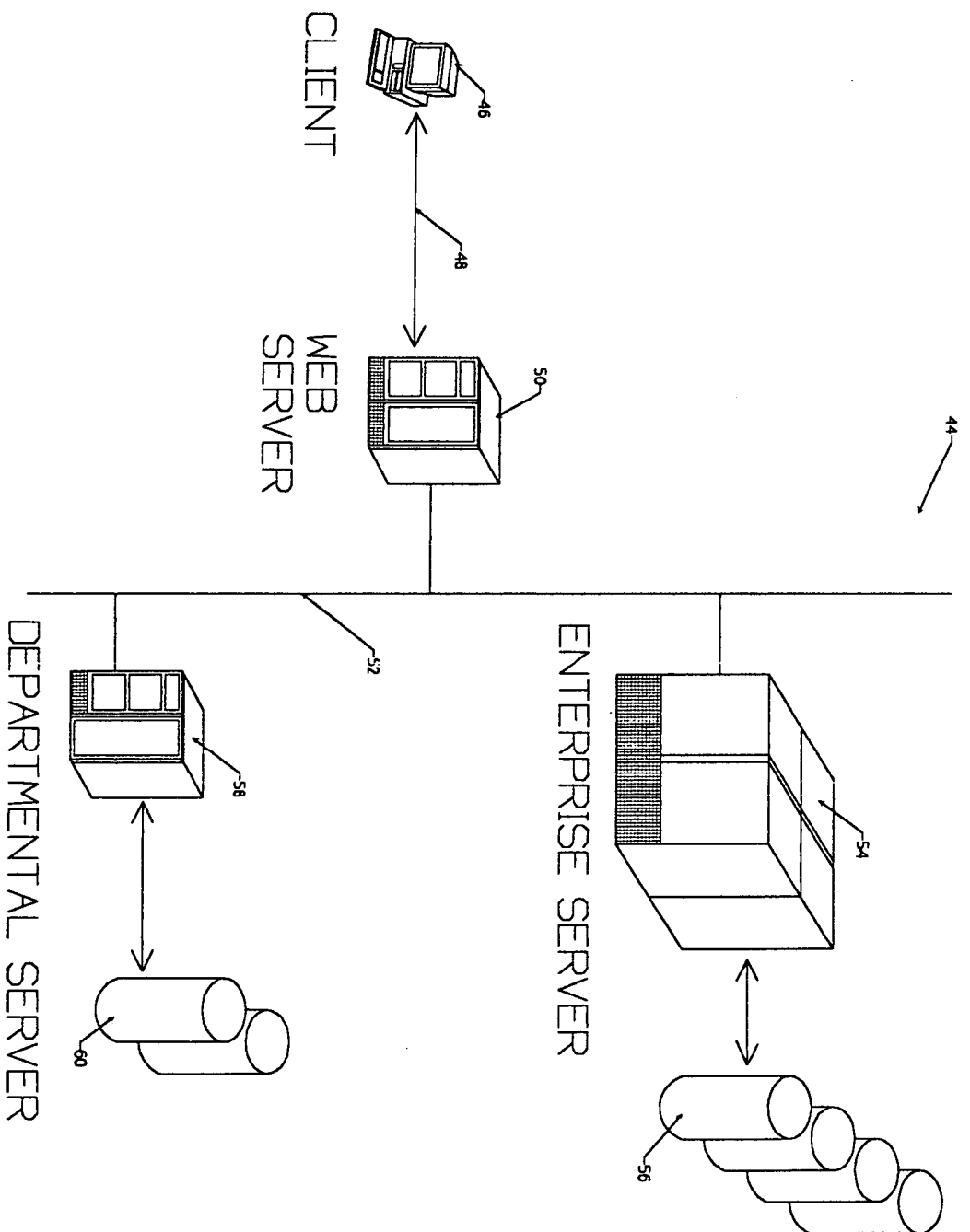


FIG. 3

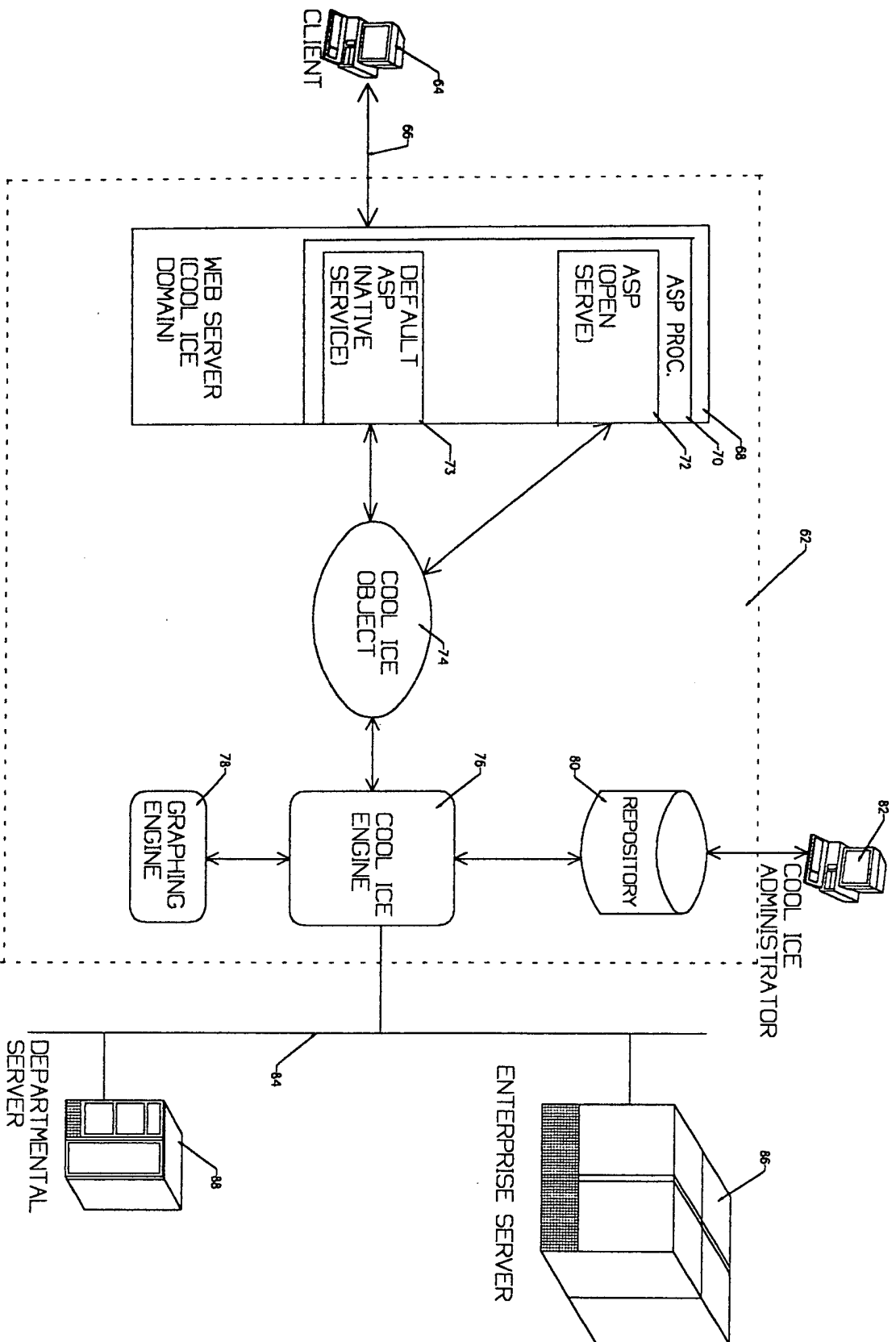


FIG. 4

Downloaded from www.worldscientific.com

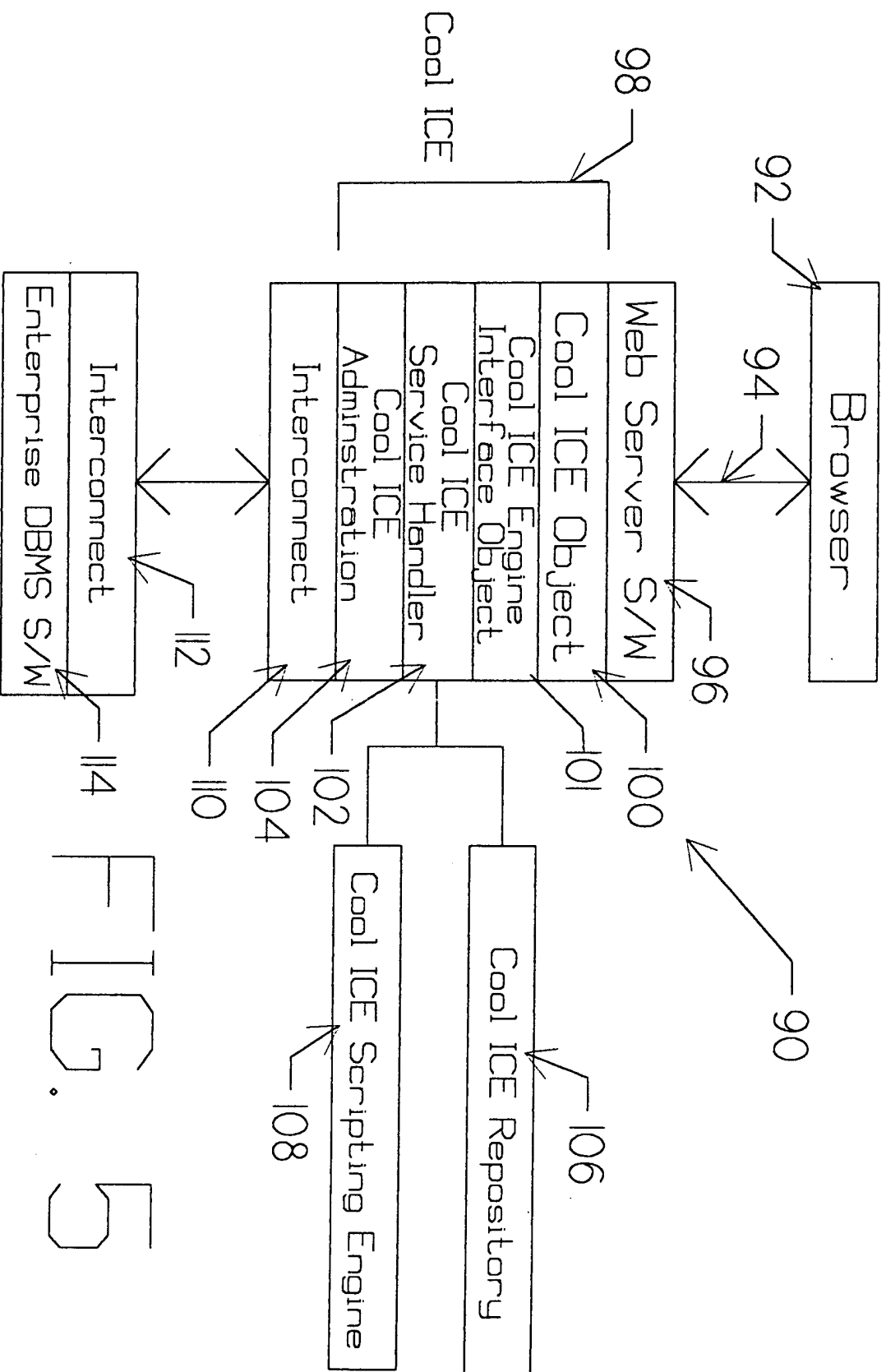


FIG. 5

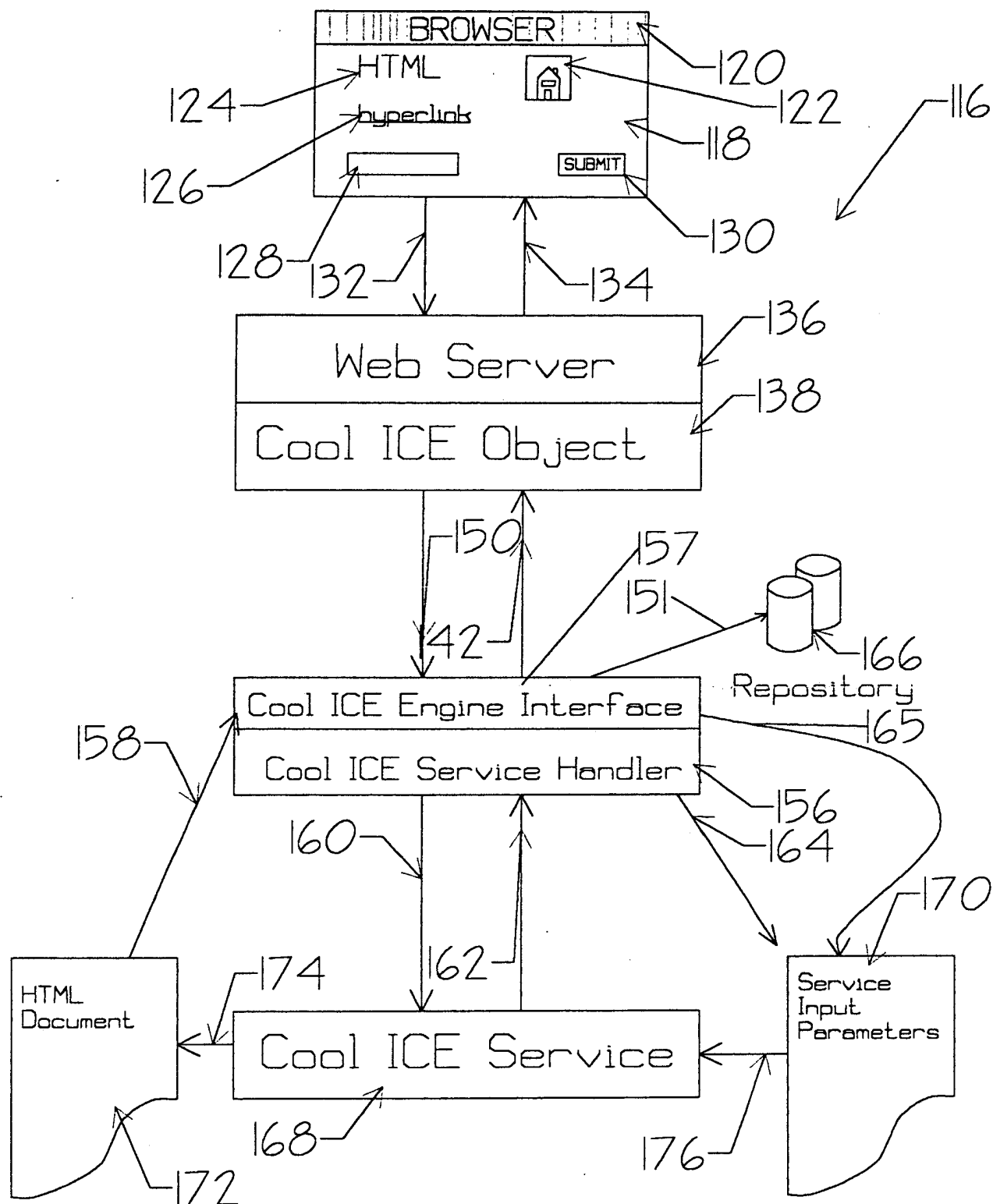


FIG. 66

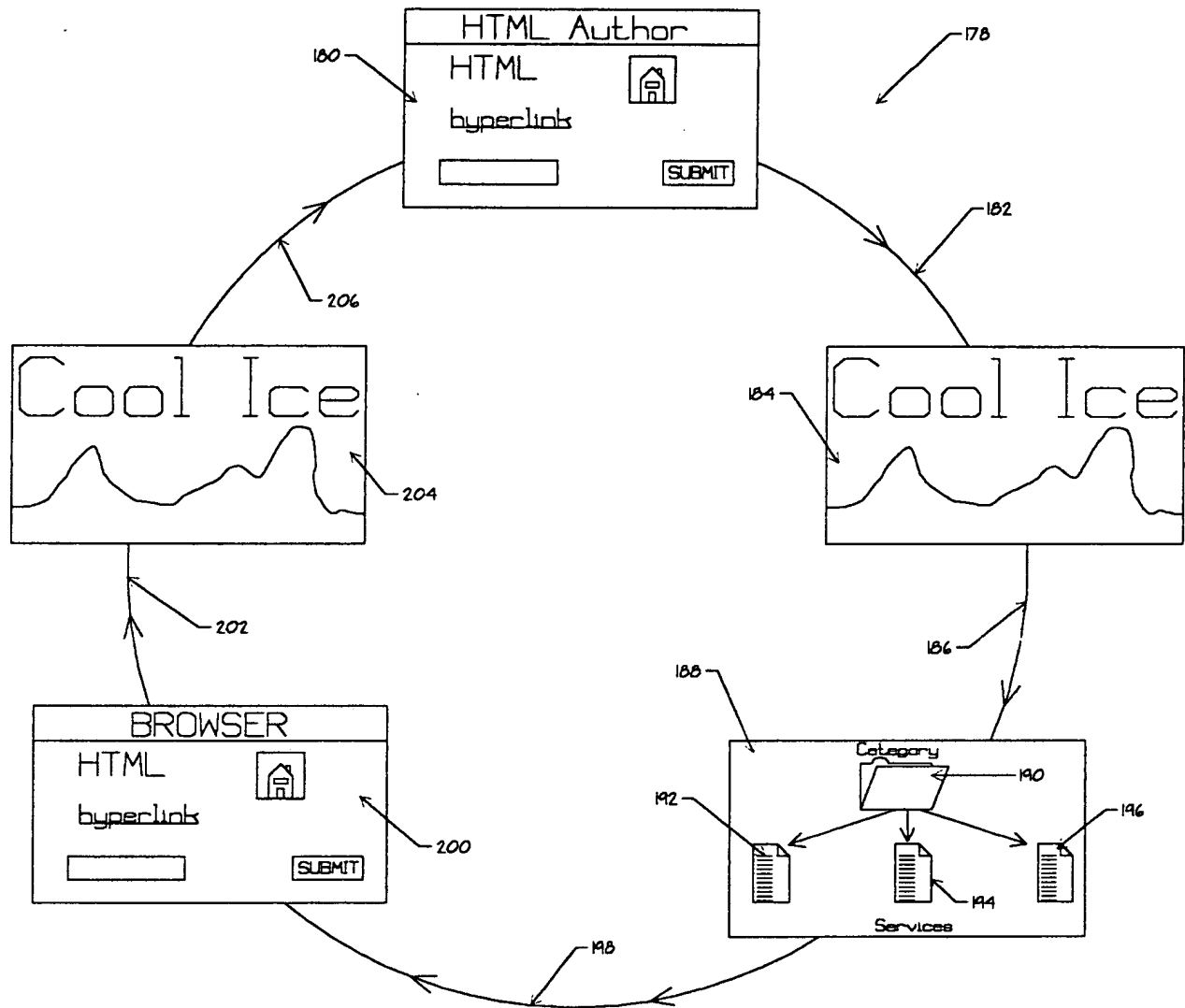


FIG. 7

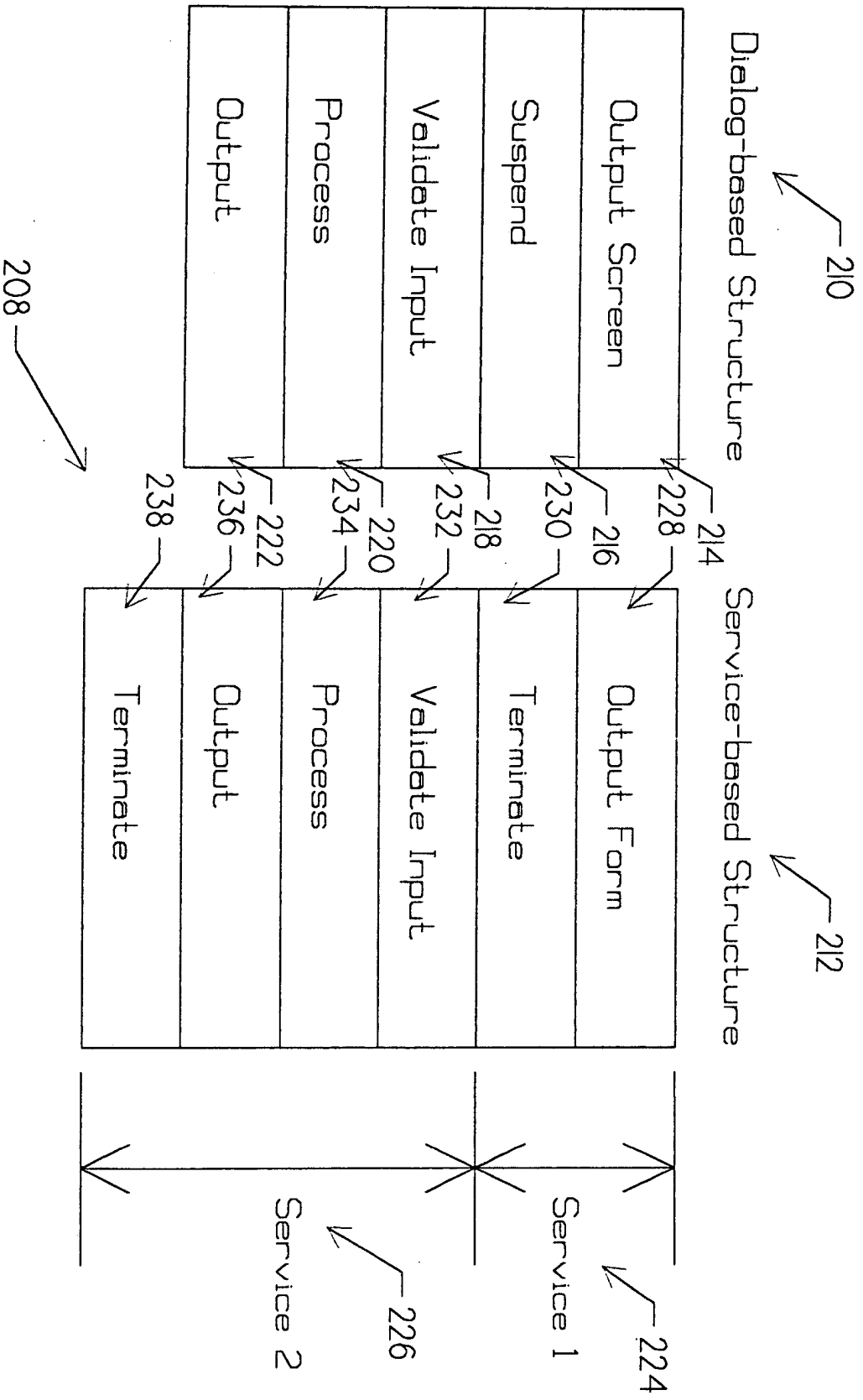


FIG. 8

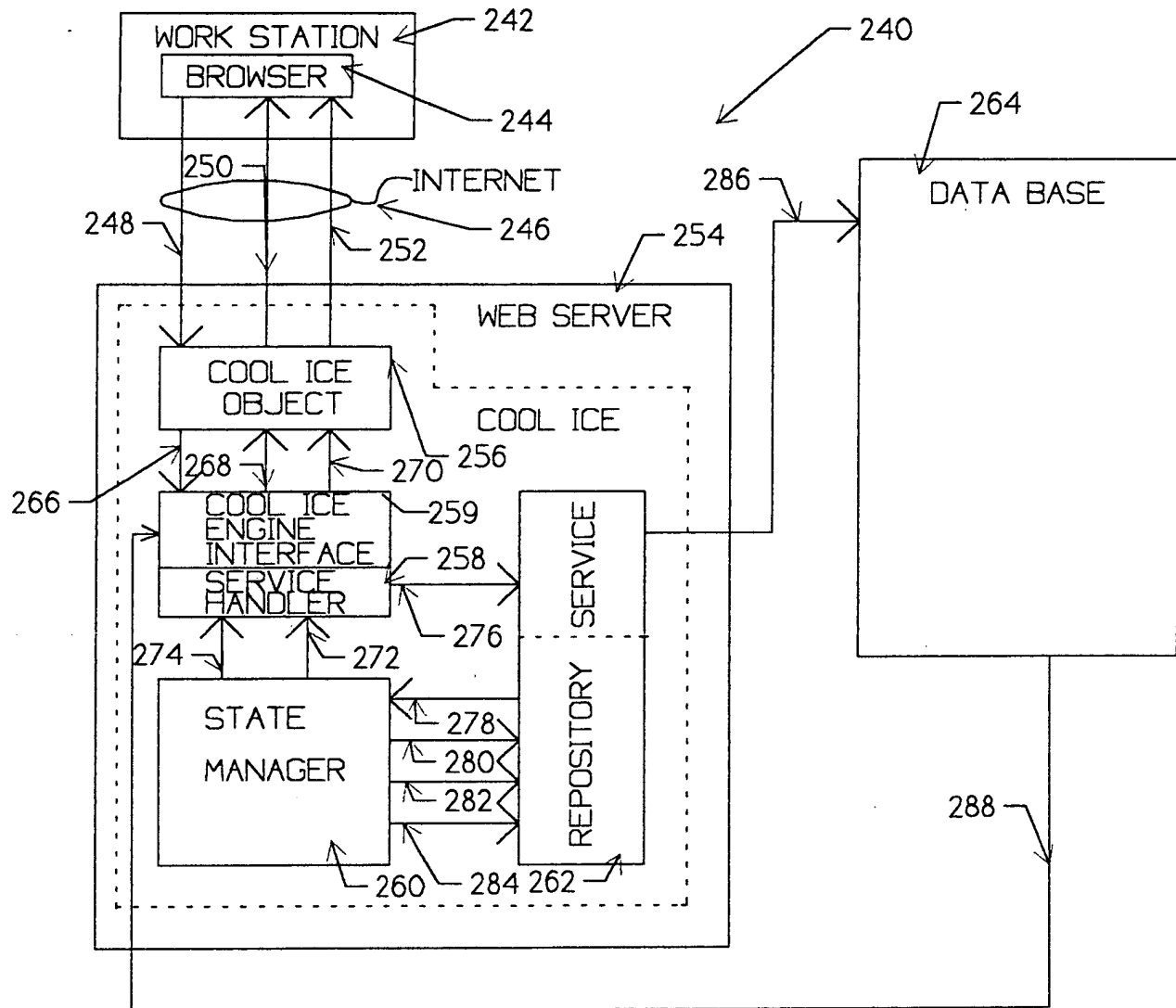
[illegible]

FIG. 9

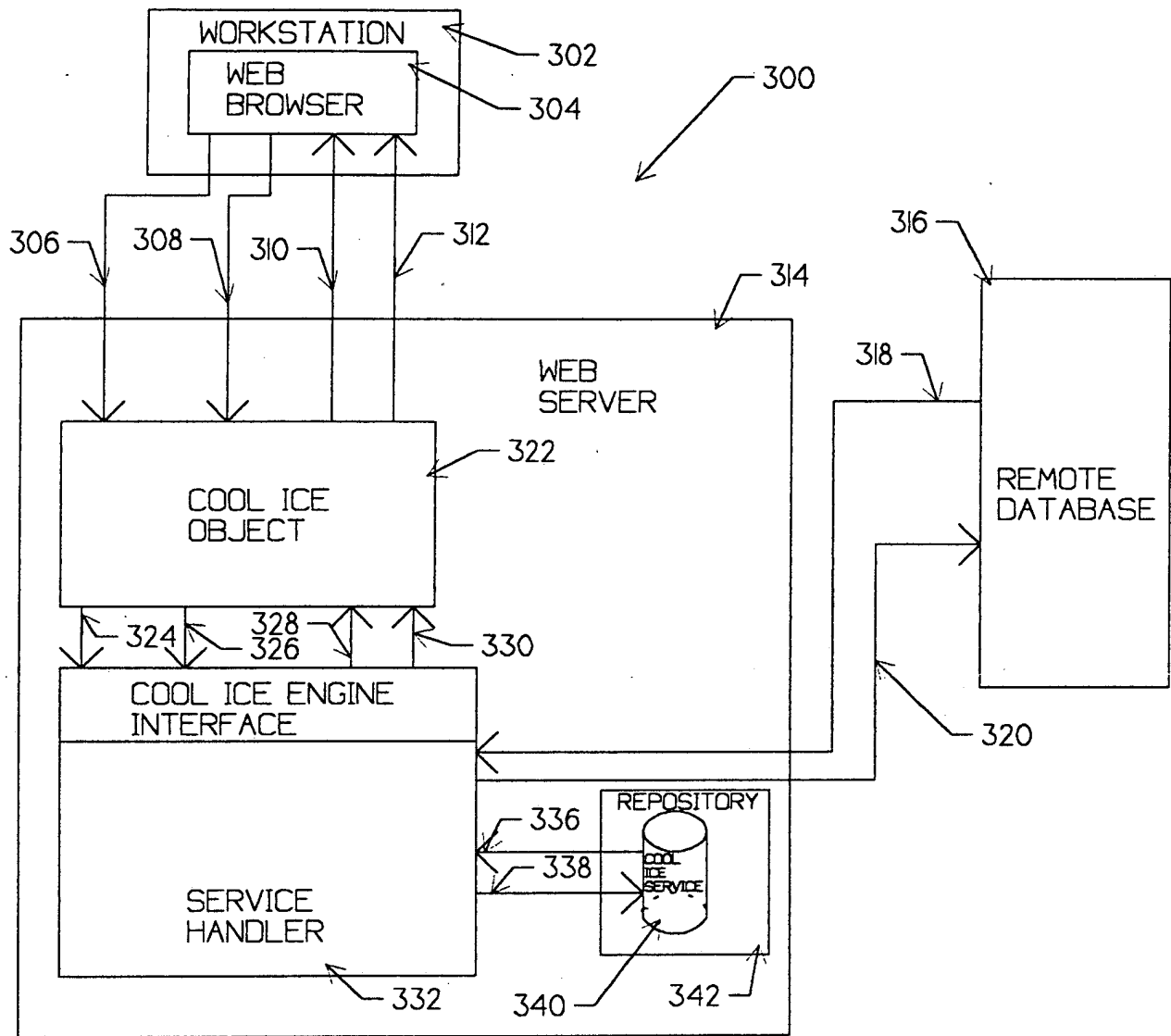


FIG. 10

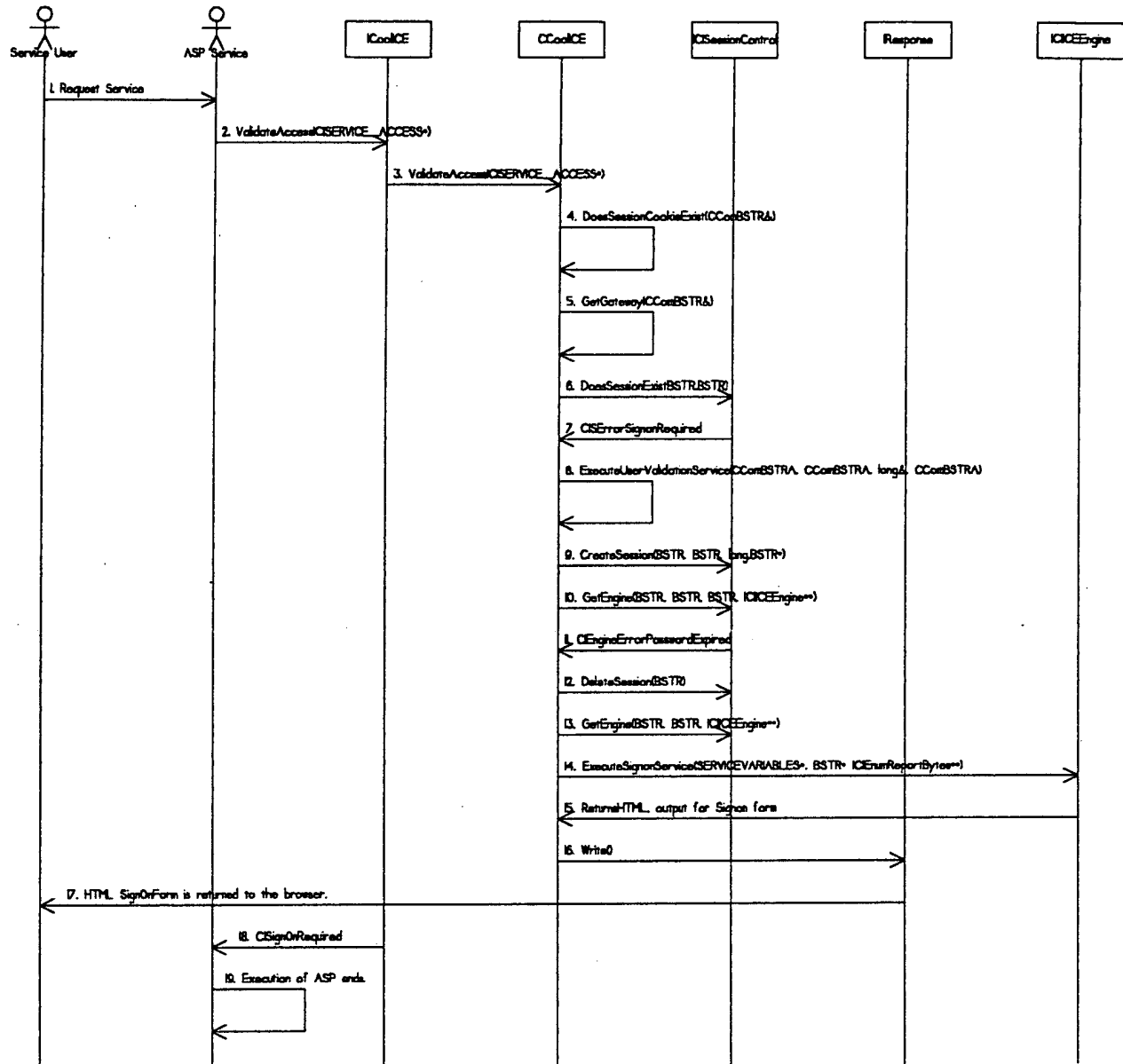


FIG. II

MESSAGE #	DESCRIPTION
1.	The Service User will make a request for an ASP page from a browser.
2.	The ASP is executed which will create a CoolICE object and call the <code>ValidateAccessMethod</code> . This method is intended to validate that the Service User does have access to the ASP being executed.
3.	
4.	Call the helper method <code>DoesSessionCookieExist</code> to determine if the <code>CSSESSIONID</code> cookie exists in the <code>Request.Cookies</code> collection. In this sequence diagram assume <code>S_FALSE</code> is returned.
5.	<code>GetGateway</code> is called to extract the Gateway Name from the ASP <code>ServerVariables</code> Collection of the ASP Request object. The <code>PATH_INFO</code> variable will return the part of the URL after the server name but before any query string. The gateway name would be the first directory in the <code>PATH_INFO</code> . For example: URL Request: <code>http://MyServer/CoolICE/abc.asp</code> <code>PATH_INFO/CoolICE/abc.asp</code> Gateway: <code>CoolICE</code>
6.	<code>DoesSessionExist0</code> is called to determine if a the HTML SignOn form needs to be processed. The <code>btnSessionID</code> parameter is set to an empty BSTR. The <code>btnGatewayName</code> parameter is the value returned by <code>GetGateway0</code> .
7.	<code>DoesSessionExist0</code> has determined that a <code>CSSESSION</code> object does not exist and a signon is required. The <code>CSSErrorSignonRequired</code> HRESULT from <code>DoesSessionExist0</code> is described in sequence diagrams SC02, SC04, SC05, and SC07.
8.	Call <code>ExecuteUserValidationService</code> to process the SignOn form input fields. In this sequence diagram assume <code>S_OK</code> is returned which indicates that a UserID, Department, and Password are returned. Optionally, a New Password may also be returned.
9.	<code>CreateSession0</code> is called to create a <code>CSSESSION</code> object. The parameters are set as follows: - <code>btnGatewayName</code> is the value returned by <code>GetGateway0</code> - <code>btnUserID</code> is the value of the <code>btnUserID</code> parameter from the call to <code>ExecuteUserValidationService0</code> - <code>btnPassword</code> is the value of the <code>btnPassword</code> parameter from the call to <code>ExecuteUserValidationService0</code> - <code>btnDepartment</code> is the value of the <code>btnDept</code> parameter from the call to <code>ExecuteUserValidationService0</code> - <code>btnSessionID</code> is the address of a local variable. In order to validate that the Service User has access to the ASP, a CoolICE engine is required. In addition, if the Service User does have access, then the CoolICE engine will be needed to allow the ASP to execute additional CoolICE services. Therefore, <code>ICSSESSIONControlGetEngine0</code> is called to access an instance of a CoolICE engine that is managed by a Connection Pool. The <code>btnSessionID</code> parameter is a unique identifier for the Service User. This identifier is returned by the <code>ICSSESSIONControlCreateSession0</code> method. The <code>btnGatewayName</code> parameter is the value returned by <code>GetGateway0</code> . The <code>btnNewPassword</code> parameter is the value of the <code>btnNewPassword</code> parameter returned by <code>ExecuteUserValidationService0</code> . In most cases, this parameter will be an empty string, except when the current password has expired, and a new password was specified.
10.	The call to <code>GetEngine</code> has returned the HRESULT value <code>CSEngineErrorPasswordExpired</code> which indicates that the specified password has expired. A special SignOn form must now be deployed which allows the user to change their password.
11.	Since an Error was returned by <code>GetEngine</code> , it is necessary to delete the <code>CSSESSION</code> object created in step 9 above. Pass the <code>btnSessionID</code> value that was returned by <code>CreateSession0</code> .
12.	It is necessary to get a Cool ICE Engine so that the SignOn service can be executed. In this case, a blank <code>SessionID</code> should be specified so that the default <code>user-id/department/password</code> will be used to sign on to the Cool ICE engine. The <code>btnSessionID</code> parameter should be a blank string so that the default <code>user-id/department/password</code> will be used to sign on to the Cool ICE engine. The <code>btnGatewayName</code> parameter is the value returned by <code>GetGateway0</code> . The <code>btnNewPassword</code> parameter should be a blank string since we know that a new password has not been specified.
13.	The <code>ExecuteSignOnService0</code> method is called to cause the SignOn service to be executed. This service will return a HTML form which has the user-id and department prefilled. A field for the old password and new password must be filled in by the user. The input parameters 'user' and 'dept' must be passed to the SignOn service when a password has expired. The parameters are passed in the <code>BrowserInput</code> section of the <code>SERVICEVARIABLES</code> structure. The values of these parameters are the <code>btnUserID</code> and <code>btnDept</code> parameters returned by <code>ExecuteUserValidationService0</code> .
14.	The SignOn service returns a HTML form that prompts the user for the old password and new password. The user-id and department number fields are prefilled and read-only.
15.	The HTML output for the SignOn service is passed to the <code>Write0</code> method of the ASP Response object.
16.	The ASP Response object will send the HTML output for the SignOn form to the browser.
17.	<code>ValidateAccess0</code> returns a <code>CSSignOnRequired</code> status back to the ASP. This will cause the ASP to terminate.
18.	Execution of the ASP is terminated due to the <code>CSSignOnRequired</code> status being returned from <code>ValidateAccess0</code> .

FIG. 12

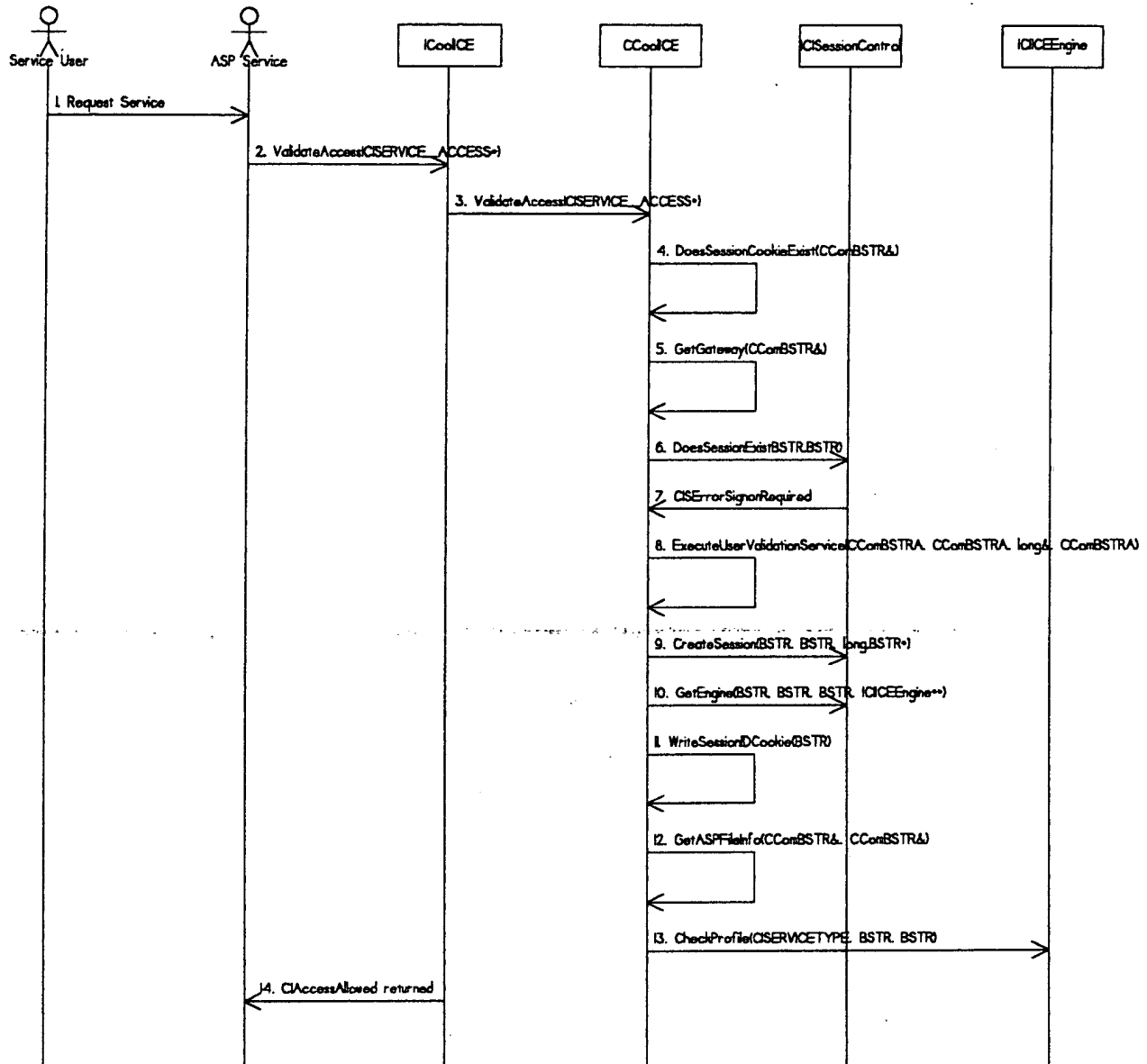


FIG. 13

MESSAGE •	DESCRIPTION
1.	The Service User will make a request for an ASP page from a browser.
2.	The ASP is executed which will create a CoolICE object and call the <code>ValidateAccessMethod</code> . This method is intended to validate that the Service User does have access to the ASP being executed.
3.	
4.	Call the helper method <code>DoesSessionCookieExist</code> to determine if the <code>CSESSIONID</code> cookie exists in the <code>Request.Cookies</code> collection. In this sequence diagram assume <code>S_FALSE</code> is returned.
5.	<p><code>GetGateway</code> is called to extract the Gateway Name from the ASP <code>ServerVariables</code> Collection of the ASP Request object. The <code>PATH_INFO</code> variable will return the part of the URL after the server name but before any query string. The gateway name would be the first directory in the <code>PATH_INFO</code>.</p> <p>For example:</p> <p>URL Request: <code>http://MyServer/CoolICE/abc.asp</code> <code>PATH_INFO</code>: <code>/CoolICE/abc.asp</code> Gateway: <code>CoolICE</code></p>
6.	<p><code>DoesSessionExist0</code> is called to determine if a the HTML SignOn form needs to be processed.</p> <p>The <code>bstrSessionID</code> parameter is set to an empty BSTR.</p> <p>The <code>bstrGatewayName</code> parameter is the value returned by <code>GetGateway0</code>.</p>
7.	<code>DoesSessionExist0</code> has determined that a <code>CCSession</code> object does not exist and a signon is required. The <code>CSCErrrorSignonRequired HRESULT</code> from <code>DoesSessionExist0</code> is described in sequence diagrams SC02 SC04, SC05, and SC07.
8.	Call <code>ExecuteUserValidationService</code> to process the SignOn form input fields. In this sequence diagram assume <code>S_OK</code> is returned which indicates that a <code>UserID</code> , <code>Department</code> , and <code>Password</code> are returned. Optionally, a <code>New Password</code> is returned.
9.	<p><code>CreateSession0</code> is called to create a <code>CCSession</code> object.</p> <p>The parameters are set as follows:</p> <ul style="list-style-type: none"> -<code>bstrGatewayName</code> is the value returned by <code>GetGateway0</code> -<code>bstrUserID</code> is the value of the <code>bstrUserID</code> parameter from the call to <code>ExecuteUserValidationService0</code> -<code>bstrPassword</code> is the value of the <code>bstrPassWd</code> parameter from the call to <code>ExecuteUserValidationService0</code> -<code>nDepartment</code> is the value of the <code>nDept</code> parameter from the call to <code>ExecuteUserValidationService0</code> -<code>pbstrSessionID</code> is the address of a local variable.
10.	<p>In order to validate that the Service User has access to the ASP, a CoolICE engine is required. In addition, if the Service User does have access, then the CoolICE engine will be needed to allow the ASP to execute additional CoolICE services.</p> <p>Therefore, <code>ICSessionControl::GetEngine0</code> is called to access an instance of a CoolICE engine that is managed by a Connection Pool.</p> <p>The <code>bstrSessionID</code> parameter is a unique identifier for the Service User. This identifier is returned by the <code>ICSessionControl::CreateSession0</code> method.</p> <p>The <code>bstrGatewayName</code> parameter is the value returned by <code>GetGateway0</code>.</p> <p>The <code>bstrNewPassword</code> parameter is the value of the <code>bstrNewPassword</code> parameter returned by <code>ExecuteUserValidationService0</code>. In most cases, this parameter will be an empty string, except when the current password has expired, and a new password was specified.</p>
11.	Call <code>WriteSessionIDCookie</code> to write the <code>SessionID</code> value returned by <code>ICSessionControl::CreateSession</code> out to the browser as the <code>CSESSIONID</code> cookie.
12.	The helper method <code>GetASPFileInfo</code> is called to retrieve the virtual directory alias name and the file name of the ASP.
13.	The <code>ICICEEngine::CheckProfile0</code> method is called to verify that the user, as known to the CoolICE engine, does have access to the ASP.
14.	The <code>ValidateAccess0</code> method will return <code>CAccessAllowed</code> status indicating that the Service User does have access, therefore, the execution of the ASP can continue.

FIG. 14